Amendments to the Claims

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Currently amended) A server method for converting Java-based objects ~~of a first type~~ into JavaScript-based objects ~~of a second type~~, the method comprising:

a) identifying one or more Java-based object classes ~~of the first type~~;

b) determining instance data based on the Java-based object classes ~~of the first type~~;

c) introspecting each Java-based object class ~~of the first type~~;

d) automatically creating an artifact representing a software model describing the instance data, the artifact mirroring the Java-based object hierarchy; ~~and~~

e) generating one or more converters for each Java-based object class, each converter being based on the artifact and ~~is~~ configured for receiving the instance data and generating JavaScript code for recreating ~~the~~ an instance from the classes as JavaScript-based objects ~~of the second type~~, for display on a browser client of the server;

wherein the converters together form a compiled recursive descent parser, synching the object hierarchies by maintaining rules from the instance data that enforce Java modeling constraints not found in JavaScript; and

f) invoking the converters at runtime to directly generate JavaScript code for projecting onto the browser client.

2. (Canceled)

3. (Currently amended) The method of claim 1 wherein the object classes comprise JavaBeans and wherein the converters are invoked at runtime to generate JavaScript code required to recreate instance data from the JavaBeans into JavaScript objects in the browser.

4. (Currently amended) The method of claim 3 ~~further comprising~~ wherein determining, for each property in a class, comprises determining whether the property is an attribute.

5. (Currently amended) The method of claim 4 ~~further comprising~~ wherein invoking comprises outputting a JavaScript code for the value of the attribute.

6. (Currently amended) The method of claim 4 further comprising wherein creating the artifact comprises creating a buffer to hold an exported object value or values if the property is an array.

7. (Original) The method of claim 6 further comprising creating an array to hold object IDs for the referenced objects for each object in the array.

8. (Original) The method of claim 7 further comprising computing a signature of the object if the object is in a map.

9. (Original) The method of claim 8 further comprising: calling an object; generating an export ID; outputting the JavaScript code to declare the object called as the converter for the object; and adding the ID to the array.

10. (Original) The method of claim 9 further comprising outputting a JavaScript array to hold the exported IDs.

11. (Currently amended) The method of claim 9 further comprising computing the signature of the object; checking ~~check~~ the ~~Map~~ map for the signature; and if the object is already in the map; retrieving ~~get~~ its export ID.

12. (Original) The method of claim 9 further comprising: calling the converter for the object; and outputting the buffer for the exported object values.

13. (Currently amended) An information processing system comprising:

a web server comprising one or more Java objects; ~~and~~

a web browser that displays JavaScript objects;

a network interface between the web server and the web browser; and

a web server program for converting the Java ~~Objects~~ objects to JavaScript objects; wherein the program comprises instructions for:

a) identifying the one or more Java object ~~object~~ classes ~~of a first type~~;

b) determining instance data based on the Java object classes ~~of the first type~~;

c) introspecting each Java object class ~~of the first type~~;

d) automatically creating an artifact representing a software model describing the instance data, the artifact mirroring the Java-based object hierarchy; ~~and~~

e) generating one or more converters for each Java object class, each converter being based on the artifact and [[is]] configured for receiving the instance data and generating JavaScript code for recreating ~~the~~ an instance from the classes as JavaScript objects ~~of a second type~~, for display on a browser client of the web server;

wherein the converters together form a compiled recursive descent parser, synching the object hierarchies by maintaining rules from the instance data that enforce Java modeling constraints not found in JavaScript; and

f) invoking the converters at runtime to directly generate JavaScript code for projecting onto the browser client.

14. (Currently amended) A computer readable medium comprising program instructions for:

a) identifying one or more Java object classes from a web server;

b) determining instance data based on the Java object classes ~~of the first type~~;

c) introspecting each Java object class ~~of the first type~~;

d) automatically creating an artifact representing a software model describing the instance data, the artifact mirroring the Java-based object hierarchy; ~~and~~

e) generating one or more converters for each Java object class, each converter being based on the artifact and [[is]] configured for receiving the instance data and generating JavaScript code for recreating ~~the~~ an instance from the classes as JavaScript objects ~~of a second type~~, for display on a browser client of the web server;

wherein the converters together form a compiled recursive descent parser, synching the object hierarchies by maintaining rules from the instance data that enforce Java modeling constraints not found in JavaScript; and

f) invoking the converters at runtime to directly generate JavaScript code for projecting onto the browser client.